U*F clustering: a new performant "cluster-mining" method based on segmentation of Self-Organizing Maps

Fabien Moutarde, Alfred Ultsch

U*F clustering: a new performant "cluster-mining" method based on segmentation of Self-Organizing Maps

Fabien Moutarde[1] and Alfred Ultsch[2]

[1] Ecole des Mines de Paris
60 Bd Saint-Michel
F-75272 Paris Cedex 06, France
Fabien.Moutarde@ensmp.fr

[2] Databionics Research Lab
Dept of Computer Science, University of Marburg
Hans-Meerwein-Straße, D-35032 Marburg, Germany
ultsch@informatik.uni-marburg.de

Abstract – *In this paper, we propose a new clustering method consisting in automated "flood-fill segmentation" of the U*-matrix of a Self-Organizing Map after training. Using several artificial datasets as a benchmark, we find that the clustering results of our U*F method are good over a wide range of critical dataset types. Furthermore, comparison to standard clustering algorithms (K-means, single-linkage and Ward) directly applied on the same datasets show that each of the latter performs very bad on at least one kind of dataset, contrary to our U*F clustering method: while not always the best, U*F clustering has the great advantage of exhibiting consistently good results. Another advantage of U*F is that the computation cost of the SOM segmentation phase is negligible, contrary to other SOM-based clustering approaches which apply $O(n^2 \log n)$ standard clustering algorithms to the SOM prototypes. Finally, it should be emphasized that U*F clustering does not require a priori knowledge on the number of clusters, making it a real "cluster-mining" algorithm.*

Key words – Self-Organizing Maps, clustering, SOM segmentation, U-matrix, data-mining.

1 Introduction

Self-organizing feature maps (SOM) may be regarded as self-organized, topology-preserving projections of high dimensional data onto a two-dimensional map [2]. This map provides a very useful and directly interpretable view of some characteristics of the analysed dataset, in particular its cluster structure [10]. On top of this ordered floor space, the U-matrix (first introduced in [7]) gives insights into the local distance structures of the data set: U-matrix visualization of trained SOM has now been for some time in the SOM community a commonly used and powerful tool for examining internal data structure of high-dimensional datasets. Most visual or algorithmic segmentations of large SOMs was done on this representation of the map (see for instance [11], [12]). This had motivated a first approach of semi-automated segmentation based on a flood-fill algorithm applied to U-matrix, which has recently been proposed in [3]. However, U-matrix depicts distances inside a cluster in the same manner as distances between different clusters. This may prevent the detection of clusters in some datasets. For this reason, an enhancement of the U-matrix (called U*-matrix) taking density information into account has been proposed in [6]. In the present work, a clustering algorithm based on U*-matrix, and using the "flooding" metaphor is proposed. For some critical datasets, the performance of this algorithm, that we nicknamed U*F, is compared to standard clustering algorithms.

2 U*F clustering method

2.1 U-matrix

The U-matrix has become the standard tool for the display of the distance structures of the input data on ESOM (Emergent SOM, i.e. SOM containing large enough number of neurons, typically several thousands, in order to obtain an interpretable 2D-projection of the studied dataset [4]). A U-matrix is constructed on top of a two-dimensional SOM grid. Let `n` be a neuron on the map, `NN(n)` be the set of immediate neighbors on the map, `w(n)` the weight vector associated with neuron `n`, then:

`U-height(n)` = $\sum_{m \in NN(n)}$ `d(w(n),w(m))`, where `d(x,y)` is the distance for input data space.

The U-matrix is a display of the U-heights on top of the grid positions of neurons on the map [4]. A U-matrix is usually displayed as a grey level picture [9], or as three-dimensional landscape [5].

2.2 Flood-fill segmentation of a U-matrix

The SOM segmentation algorithm proposed in [3] was a simple area-filling algorithm applied to the U-matrix of the SOM. More precisely, let *U-height(i,j)* be the U-matrix value at position (*i,j*) on the SOM grid. Then, the following region-growing algorithm was applied:

- empirically define, for each visually-identified cluster C_k, a threshold distance $dmin_k$
- start from any neuron $n_{i0,j0}$ which is clearly inside the cluster C_k
- apply to *(i0,j0, $dmin_k$, k)* the following recursive procedure:

floodFill(i, j, $dmin_k$, k) {
if (*i,j*) is inside the SOM grid range, then:
if ($n_{i,j}$ is not tagged as member of C_k) and (*U-height(i,j)* < $dmin_k$), then do:
- tag $n_{i,j}$ as member of C_k
- call *floodFill(i+1,j,$dmin_k$,k)* - call *floodFill(i-1,j,$dmin_k$,k)*
- call *floodFill(i,j-1,$dmin_k$,k)* - call *floodFill(i,j+1,$dmin_k$,k)*

}

This procedure applied to U-matrix produces good results (see [3]), as long as the U-matrix exhibits well-separated zones for each data cluster. This is not always the case in practice, especially when closely neighboring clusters have low density near their "contact zone", hence the idea of applying it to U*-matrix instead. One might also wonder if improvement could be obtained by using the actual inter-neuron distances (`d(w(`$n_{i,j}$`),w(`$n_{i+1,j}$`))`, etc...) to propagate on a different criteria in each direction, instead of using for all four directions the mean of distances to neighbors. In fact, according to our experiments, this seems to make things worse, because then frontiers are more easily crossed-over, forcing to choose a lower threshold and thus leaving the "basins" partially unfilled.

2.3 U*-matrix

In dense regions of the data space, the local distances depicted in a U-matrix are presumably distances measured inside a cluster. Such distances may be disregarded for the purpose of clustering. In thin populated regions of the data space, however, the distances matter. In this case the U-matrix heights correspond to cluster boundaries. This lead to the definition of a U*-matrix, which combines the distance-based U-matrix and a density-based P-matrix defined in [6]. The P-height of a neuron *n*, with associated weight vector *w(n)*, is defined as: *P-height(n) = p(w(n),X)* where `p(x,X)` is an empirical density estimation at position x in the space points distribution of dataset X. In principle, any of several various existing methods could be used for the estimation of density. In practice, we use the Pareto Density Estimation, which consists in counting data points inside a hypersphere centered on point `x`, and with a radius equal to the "Pareto radius" (see [6] and references therein for more details).

The U*-matrix is then derived from a U-matrix following these lines:

- when the data density around a weight vector of a neuron is equal to the average data density, the heights shown on a U*-matrix should be the same as in the corresponding U-matrix;
- when the data density around a weight vector of a neuron is big, local distances are primarily distances inside a cluster; in this case the U*-matrix heights should be low;
- when the data density around a weight vector of a neuron is lower than average, local distances are primarily distances at a border of a cluster; in this case the U*-matrix heights should be higher than the corresponding U-height.

This leads to the following formula: let *U-height(n)* denote the U-matrix value at neuron *n*, let *mean(P)* denote the mean of all *P-heights*, and *max(P)* the maximum of all P-heights, then the U*-height for neuron *n* is calculated as:

$$U^*\text{-}height(n) = U\text{-}height(n) * ScaleFactor(n), \text{ with}$$

$$ScaleFactor(n) = \frac{Pheight(n) - mean(P)}{mean(P) - \max(P)} + 1$$

This definition ensures that U*-height<U-height when P-height>mean(P) (with U*-height=0 when P-height=max(P)) which happens inside clusters, while on the contrary U*-height>U-height when P-height<mean(P) which normally happens essentially between clusters.

2.4 U*F clustering

U*F clustering is the application to the U*-matrix of an improved version of the segmentation algorithm described in §2.2. U*-heights are used instead of U-heights, and the region-growing procedure has been further automated: the threshold for stopping the region-growing process is now automatically determined by choosing the value above which the filled area suddenly grows dramatically (which reveals an overflow in a neighboring region), as shown below. The threshold value is simply determined by measuring the size (in pixels) of the grown-region for several (typically 100) evenly spaced values of threshold in the [0;1] interval, and looking for the point of maximum gradient of the regionSize=f(threshold) function. This can produce a meaningful choice of threshold value only if the map it is applied to exhibits some relatively well-defined "basins", which is generally enhanced by using U*-matrix instead of U-matrix.

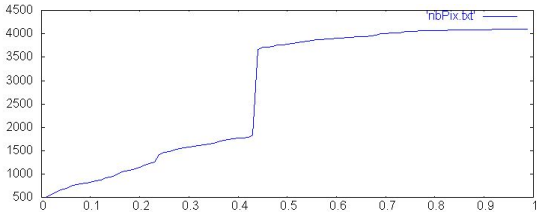

Figure 1: typical numberOfPixels=f(threshold) curve, showing the pixel number of the region grown by the flood-fill algorithm as a function of the `dmin` threshold value. On this example, the optimal threshold value is clearly identified by the large step near `dmin=0.45`.

3 Experiments

3.1 Datasets

Atom: The Atom dataset (see fig.2a) is 3D and consists in two clusters A and B of 400 points each. Cluster A fits within a sphere of radius 11.5 around the origin. Cluster B fits within a spherical shell with minimal and maximal radius 48.5 and 51.5, also centred on the origin. The minimal distance between the two subsets is far bigger than the diameter of A. This clustering problem is difficult since the clusters are not separable by any hyperplane. Cluster A is much more dense than B. The inner distances of cluster B are up to twice as big as the distances from A to B.

WingNut: The WingNut dataset (see figure 2a) consists in two symmetric data subsets of 500 points each. Each of these subsets is an overlay of equal spaced points with a grid distance of 0.2 and random points with a growing density in one corner. The data sets are mirrored and shifted such that the gap between the subsets is bigger than 0.3. Although there is a bigger distance in

between the subsets than within the data of a subset, clustering algorithms like K-means parameterized with the right number of clusters (k=2) produce classification errors.

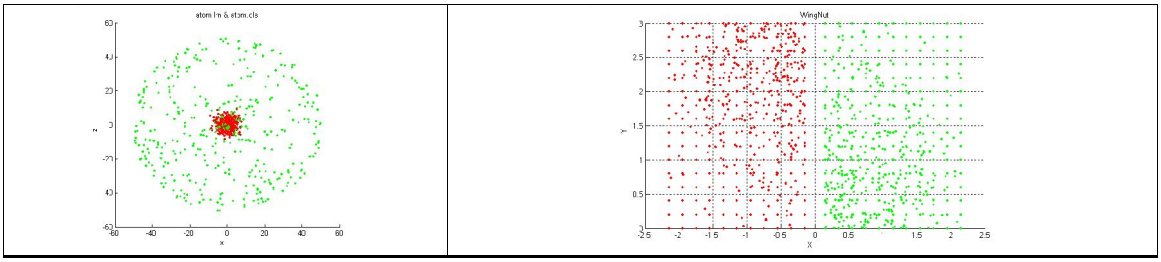

Figure 2a: some of the artificial datasets used in the experiments (left: 2D projection of Atom; right: WingNut).

Lsun: Lsun consists in three well-separated 2D clusters (two with 100 points, and one with 200 points). The inter-cluster minimum distances, however, are in the same range or even smaller than the inner-cluster mean distances.

TwoDiamonds: The TwoDiamonds dataset (see figure 2b) consists in two clusters with 300 points in each. Each cluster points are uniformly distributed within a square, and at one point the two squares almost touch (see [6]). This dataset is critic for clustering algorithms using only distances.

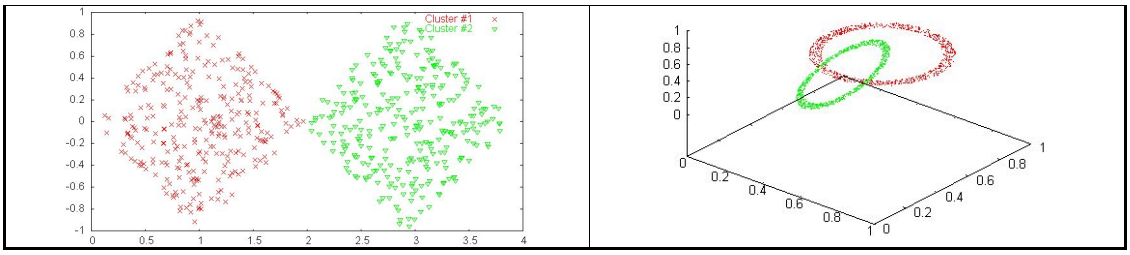

Figure 2b: some of the synthetic datasets used in the experiments (left: TwoDiamonds, right: ChainLink).

ChainLink: The ChainLink dataset (see figure 2b) has been used in [8] to show that large SOMs (ESOM) clustering is different from K-means. It consists in two tore-shaped clusters of 500 points each, which are intertwined like the links of a chain. The clusters, although well separated, are difficult to cluster since they are not separable by any linear or quadratic manifold.

3.2 U*F clustering results

For the "Atom" dataset, U*F clustering produces an absolutely perfect cluster identification, as illustrated by the confusion matrix below on the left. We show results with toroid SOM on this dataset to illustrate applicability of U*F on that kind of SOM; if planar topology is used on Atom, U*F results are still excellent, except that one of the clusters ends up split in two.

		Clusters determined by U*F method		
		1	**2**	**Total**
True clusters of Atom dataset	**1**	**400**	**0**	**400**
	2	**0**	**400**	**400**
	Total	**400**	**400**	

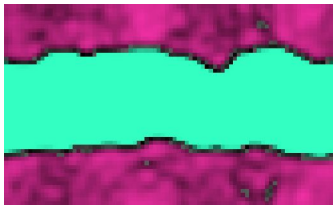

Figure 3a: Atom dataset U*-matrix segmentation determined by U*F; the upper and lower zones are the same because the map topology is toroidal.

Performance of U*F clustering on the Lsun dataset is nearly as perfect. As shown on fig.3b, the number of clearly separated zones on the U*-matrix (visually determined) is 3, which is exactly the number of true clusters. The resulting clustering for Lsun dataset is nearly perfect, except for 5 examples from true cluster #3 which are left "unclassified" (see table below).

Figure 3b: Lsun dataset U*-matrix segmentation determined by U*F method

		Clusters determined by U*F method				
		1	2	3	None	Total
True clusters of Lsun dataset	1	200	0	0	0	200
	2	0	100	0	0	100
	3	0	0	95	5	100
	total	200	100	95	5	

The performance of U*F clustering on WingNut dataset is slightly less good: as shown in the confusion matrix below, a significant proportion (9%) of examples are mistakenly left isolated in none of the clusters. However, it is important to notice that absolutely no example was assigned to the wrong cluster, and the number of clusters was very clearly and automatically identified as 2, as can be seen on figure 3c.

		Clusters determined by U*F			
		1	2	None	Total
True clusters of WingNut dataset	1	455	0	45	500
	2	0	456	44	500
	Total	455	456	89	

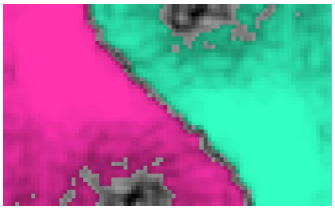

Figure 3c: WingNut dataset U*-matrix segmentation determined by U*F method.

U*F outcome on the TwoDiamonds dataset is similar: still no example placed in the wrong cluster, but 12% of the examples mistakenly left isolated in none of the clusters, as shown on the resulting confusion matrix below:

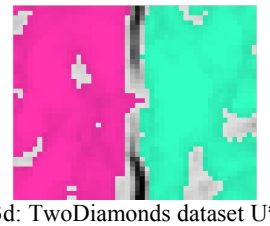

Figure 3d: TwoDiamonds dataset U*-matrix segmentation determined by U*F method

		Clusters determined by U*F			
		1	2	None	total
True clusters of TwoDiamonds dataset	1	259	0	41	300
	2	0	270	30	300
	total	259	270	71	

On the "ChainLink" dataset, U*F produces 3 regions (see figure 3e), even though visual inspection of the U*-matrix clearly suggests 2 separated regions (which is the true number of data groups). However, the "extra" region is entirely within one of the true data groups, so that the consequence is just an artificial division of one of the actual groups in two clusters. On this particular dataset, the U-matrix is in fact easier to segment than the U*-matrix (see §3.3).

Figure 3e: ChainLink dataset U*-matrix segmentation determined by U*F method.

		Clusters determined by U*F				
		1	2	3	None	total
True "clusters" of ChainLink dataset	1	471	0	0	29	500
	2	0	321	153	26	500
	total	471	321	153	55	

3.3 U*F variant

In some isolated cases, it seems that a better result can be obtained by applying U*F to the U-matrix instead of the U*-matrix. For instance, using this variant of U*F on the ChainLink dataset significantly improves the result (see figure 4 and table below).

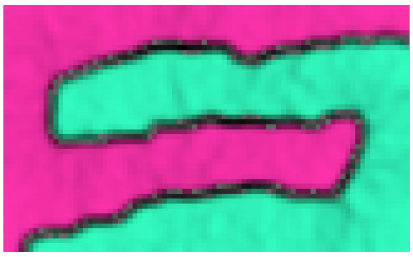

Figure 4: SOM segmentation for ChainLink when applying U*F to the U-matrix instead of the U*matrix.

		Clusters determined by U*F variant (segmentation based on U-matrix instead of U*-matrix)			
		1	2	None	total
True "clusters" of ChainLink dataset	1	500	0	0	500
	2	0	500	0	500
	total	500	500	0	

This U*F variant can also be useful for datasets for which at least one of the input component is discrete-valued. Because computation of the U*-matrix requires an evaluation of local density in the input space (see §2.3), it is not readily applicable to these kinds of datasets. However, since the U*F variant described above only requires the U-matrix, it is still possible to apply this variant for these categories of datasets, as illustrated on the following example.

The "dermatology" dataset (originating from Gazi University school of medicine and Bilkent University Computer Science department, Ankara, Turkey, and available on the machine-learning database repository of University of California at Irvine, located at URL `http://www.ics.uci.edu/~mlearn/MLSummary.html`) contains 358 examples corresponding to 6 categories of erythemato-squamous diseases. Each example is a 34-dimensional vector, ***with all-but-one components discrete-valued***. Because of this, U*-matrix, as explained above, is not readily computable for this dataset. But U-matrix can be computed, and the U*F variant applied, with the results illustrated below.

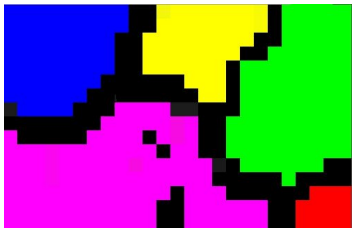

Figure 5: segmentation obtained with U*F algorithm applied to U-matrix (instead of U*-matrix) for the dermatology dataset.

	Cluster #1	Cluster #2	Cluster #3	Cluster #4	Cluster #5	Outside clusters
Psoriasis	102	-	-	-	-	9
Pytir_rubra_pilaris	-	19	-	-	-	1
Lichen_planus	-	-	70	-	-	1
Pytiriasis_rosea	-	-	-	47	-	1
Seboreic_dermatitis	-	-	-	57	-	3
Chronic_dermatitis	-	-	-	-	44	4

Note that the number of regions was visually determined by inspection of the U-matrix, and of region-growing outcome. The above table shows that the U*F algorithm applied on U-matrix produced very good results. It was not able to distinguish two of the actual categories (pytiriasis_rosea and seboreic_dermatitis, which end up in the same cluster), but there is absolutely no mixing of examples from different real categories, and only 5.4% of the examples are left outside any cluster.

3.4 Comparison with other clustering algorithms

As a comparison, we applied some standard clustering algorithms directly to the same artificial datasets. We chose on purpose three rather different types of algorithms: single-linkage and Ward clustering which are two very different kinds of hierarchical agglomerative techniques, and K-means (known for its bias towards spherical clusters). Below is a summary of the results:

Dataset	SOM topology	Single-linkage	Ward	K-means	U*F clustering
Atom	Toroidal (50x82)	**Perfect**	34 % in ***wrong*** cluster	28 % in ***wrong*** cluster	**Perfect**
Lsun	Planar (50x82)	25 % in ***wrong*** cluster	23 % in ***wrong*** cluster	28 % in ***wrong*** cluster	**No error** (but 1 % ***not in any cluster***)
WingNut	Planar (50x82)	50 % in ***wrong*** cluster	4 % in ***wrong*** cluster	4.6 % in ***wrong*** cluster	**No error** (but 9 % ***not in any cluster***)
ChainLink	Planar (50x82)	**Perfect**	23 % in ***wrong*** cluster	35 % in ***wrong*** cluster	No error (but 1 group split in 2, and ≈ 5 % ***not in any cluster***)
Two Diamonds	Planar (40x50)	50 % in ***wrong*** cluster	0.5% in ***wrong*** cluster	**Perfect**	**No error** (but 12 % ***not in any cluster***)

It can be seen that in our experiments, U*F clustering never mixed together examples from different true clusters, which was not the case for neither single-linkage, nor Ward, nor K-means clustering. On the other hand, a sometimes significant proportion of the examples were not affected to any cluster by U*F, and occasionally a true cluster ended divided in two.

4 Discussion

According to our experiments, the U*F clustering method presented in this paper generates not perfect, but consistently good clustering results. In particular, and in contrast to some common standard clustering algorithm, it rarely mixes together data points that actually belong to different true clusters. A first promising result on real data was obtained on a medical dataset with the U*F variant using only U-matrix instead of U*-matrix (see §3.3).

However more tests should now be conducted to confirm the efficiency of our U*F method, especially on various real datasets, as well as on artificial datasets where the clusters are not well separated but only form more dense areas in the data. Some very preliminary results (not yet fully analyzed in time to be formally exposed in the present paper) on the last kind of dataset give indication that U*F still works rather well on these kinds of datasets, except for a tendancy to leave "unaffected" to any cluster an important proportion of the data (in other words, it seems to identify correctly essentially the "cores" of the clusters). The two main drawbacks of U*F clustering identified so far are thus:

a) Building the U*-matrix requires the computation of local density in the input space, which makes it not very well suited for datasets with at least one discrete-valued component.
b) For several datasets, U*F appears to mistakenly leave a significant proportion of the examples isolated in none of the clusters.

However, it should be noted that for datasets corresponding to the first case, it is still possible to apply the flood-fill segmentation on the U-matrix instead of U*-matrix, and still obtain an acceptable result with this U*F variant, as illustrated in §3.3. The other identified weakness of U*F clustering can in fact be regarded as an advantage compared to other clustering algorithms which force categorization of every example in one of the clusters, sometimes leading to an important number of categorization errors.

Also, it could be argued that SOM segmentation by a classical clustering method applied to the SOM prototypes, as proposed by [12], is more mathematically sound. It would be interesting to compare the clustering results of both approaches. However, it should be noted that, as pointed out in [1], standard hierarchical clustering techniques have an over-all computational complexity of at

least $O(n^2 \log n)$ where n is the number of elements to cluster. A great advantage of our U*F method is that the computation cost of the segmentation phase is O(n) where n is the number of SOM units, so that its global complexity is essentially that of the computation of the U*-matrix (or just the U-matrix, in case the variant of U*F is used).

5 Conclusion

We have proposed a new clustering method, called U*F clustering, and based on automated "flood-fill segmentation" of U*-matrix of Self-Organizing Maps after training. It was shown by testing its clustering performance on several critical datasets that our U*F method shows consistently good clustering results. This "consistence" is in contrast with other clustering algorithms (K-means, single-linkage, and Ward) to which we compared U*F: they may sometimes perform better than U*F, but each of them performs very poorly on at least one particular kind of datasets. Moreover, our U*F has the following advantages:

- when the categorization is not perfect, examples are left "isolated" rather being attributed to the wrong cluster ;
- no a priori hypothesis for the number of clusters is required ;
- the global computation cost is essentially equal to that of the computation of the U*-matrix, in contrast with other approaches applying standard clustering algorithm to SOM units.

In conclusion, U*F clustering method seems to be a very performant alternative to usual clustering algorithms (such as K-means, single-linkage, Ward, etc...), and a promising data-mining tool for "blind cluster discovery".

References

[1] M. Dash and H. Liu (2001), Efficient hierarchical clustering algorithms using partially overlapping partitions, *Lecture Notes in Computer Science*, **vol. 2035**, p. 495-507.

[2] T. Kohonen (1982), Self-Organized formation of topologically correct feature maps, *Biological Cybernetics*, **vol. 43**, p.59-69.

[3] D. Opolon & F. Moutarde (2004), Fast semi-automatic segmentation algorithm for Self-Organizing Maps, In *Proc. of ESANN'2004, Bruges, 28-30 avril 2004*, p. 507-512.

[4] A. Ultsch (1992), Self-Organizing Neural Networks for Visualization and Classification, In *Proc. Conf. Soc. for Information and Classification, Dortmund (Germany), April 1992.*

[5] A. Ultsch (2003), Maps for the Visualization of high-dimensional Data Spaces, In *Proc. WSOM'03, Kyushu (Japan)*, p. 225-230.

[6] A. Ultsch (2003), U*-Matrix: a Tool to visualize Clusters in high dimensional Data, In *Research report Dept. of Mathematics and Computer Science, University of Marburg (Germany)*, No. 36.

[7] A. Ultsch & H.P. Siemon (1990), Kohonen's Self Organizing Feature Maps for Exploratory Data Analysis, In *Proc. Intern. Neural Networks Conf. (INNC'90), Dortrecht (Netherlands),* Kluwer Academic Press, Paris, p. 305-308.

[8] A. Ultsch & C. Vetter (1994), Self-Organizing-Feature-Maps versus statistical clustering methods: a benchmark. *FG Neuroinformatik & Kuenstliche Intelligenz, University of Marburg, Research Report 0994.*

[9] J. Vesanto et al. (1999), Self-organizing map in Matlab: the SOM toolbox, In *Proceedings of the Matlab DSP Conference*, Espoo, Finland, November 1999, p. 35-40.

[10] J. Vesanto (1999), SOM-based data visualization methods, *Intelligent Data Analysis*, **vol. 3 (2)**.

[11] J. Vesanto (2000), Using SOM in data mining, *Licentiate thesis*, Helsinki University of Technology.

[12] J. Vesanto & E. Alhoniemi (2000), Clustering of the Self-Organizing Map, *IEEE Transactions on Neural Networks*, **vol. 11 (3)**.